www.didel.com/OledFunctions.pdf

OledPix and OledMap functions

Oled SSD1306 software

The soft is modular and can be as small as 500 bytes; the single .h file can be split, stripped or completed according to the application. It has been written in simple C and tested on Arduino IDE, not using Arduino functions.
There are two possibilities for I2C transfers, see www.didel.com/OledI2C.pdf
There are two versions with or without buffer, optimized for the cheap SSD1306, 128x64 pixels, see www.didel.com/OledLib.pdf
Sources can be found on https://github.com/nicoud/Oled

OledI2C.h	*on I2C pins* *TWI on AVR 328*	**OledI2Cbb.h**	*on any 2 pins* *bit-bang 100kb/s*
OledMap.h	*need 1k byte buffer*	**OledPix.h**	*no buffer lib*

OledMap and OledPix consist of several files, some are common, not depending on hardware.

OledMap	Common	**OledPix**
OledControlMap.h write on map, init and transfer map on Oled		OledControlPix.h init and write on Oled
	OledGenc .h character generator and sprites	
OledCarMap.h char and text written to buffer		OledCarPix.h char and text sent to Oled
OledBigMap.h double size char and number		OledBigPix.h double size char and number
	OledNum.h 8 and 16 bits Hex and Dec numbers	
OledGraMap.h Dot(x,y) and basic graphic		OledGraPix.h Dot(x,y) modify 8 pixels
	OledLineCircle.h not usable with OledGraPix	

Why is OledLineCircles.h not usable with OledGraPix?
Circle was drawn first. The missing dots have been calculated and transferred. On the vertical part of the circle, there has been several dots in the same oled byte.
Writing a new dot means writing a byte that erases previous dots.
Diagonal lines have erased several circle dots, on the 8-bit strip of a line.
On a buffer in RAM, read-modify-write is performed and previous dots are not lost.

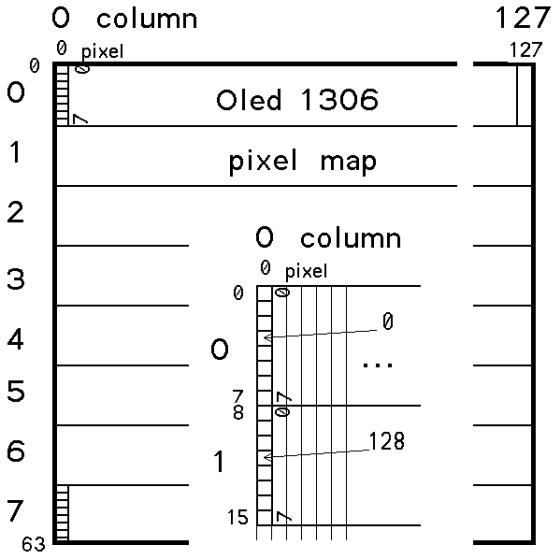
List of functions

Library files	Functions	Sizes Arduino
OledI2C.h OledI2Cbb.h	Stop(); Start(); Write();	~150
OledControlPix.h OledControlMap.h	Cmd (c); SetupOled(); Clear(); Licol(li,co); CopybMap(bmap); Show(); *(Map only)*	~300
OledGenc.h	Ascii 32-127 Smile(); Sad();	600
OledCar.h	Car(cc); Text(); Error();	~250
OledNum.h	Bin8(); Hex8(); Hex16(); Dec8(); Dec9999();	~400
OledBig.h	Big(); BigBin8(); BigHex8(); BigHex16(); BigDec8(); BigDec9999();	~500
OledGra.h	Dot(x,y); DDot(x,y); Vline(x); Hline(y);	~100
OledLineCircle.h	Line(x0,y0,x1,y1); Circle(x,y,r);	~400

Cmd (0xA4); // A7 inverse A6 direct A5 tout blanc A4 normal
Cmd (0xda);Cmd (0x02); //DoubleHmode

Comments

Most functions are evident if the user has already programmed a display.
Oled bitmap has the same structure as SSD1306 pixel map (this is not the case with GFX).
Text must be aligned on one of the 8 byte lines.
Licol (line,col) is used to position the pointer before writing text or numbers.
byte line = 0..7; byte col = 0..127;
Double size numbers cannot start at line 0.
Dec9999(); is a little bit unusual. It converts hex integers up to 0x270F. This avoid one more digit and is good for showing 10-bit A/D converters values.
Graphic objects use x y coordinates.
byte x = 0..127 byte y = 0..63
DDot(x,y) is 2 dots on top of each other, to make the two tracks of a "scope" more easy to recognize.



Text and Sprites

Texts must be defined as tables, e.g. byte sayHello[]={"Hello"}; is called with the function Text():
Text(sayHello);
Sprites are defined as a list of bytes corresponding to the vertical bytes of the display. Sprites may use several lines, but they will have to be handled as several sprites to be positionned accordingly.
User's sprites are defined as a table, and called with MySprite(mySprite).
Character generators and predefined sprites are stored in program memory (with PROGMEM) .
More details of functions and program examples can be found on www.didel.com/Oled1306.pdf.

		
Demo of the moving average on noisy data	Debugging screen on new hardware.	Record: 512 bytes on a primitive micro

jdn 170611